InvenSense Inc.
1197 Borregas Ave., Sunnyvale, CA 94089 U.S.A.
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104
Website: www.invensense.com

Document Number:
Revision:
Release Date:

Motion Sensors Introduction

A printed copy of this document is
NOT UNDER REVISION CONTROL
unless it is dated and stamped in red ink as,
"REVISION CONTROLLED COPY."

TABLE OF CONTENTS

1. REVISION HISTORY 3
2. PURPOSE 3
3. INTRODUCTION TO MOTION SENSORS 4
4. INTRODUCTION TO SENSOR FUSION 5
5. MOTION INTERFACE APPLICATIONS 7
6. REFERENCE 8
APPENDIX: DIFFERENT REPRESENTATIONS OF DEVICE ORIENTATION 9

1. Revision History

Revision Date	Revision	Description
06/26/2012	1.0	Document created

2. Purpose

This document provides a quick overview of the basics of motion sensors, sensor fusion, and a list of applications that would benefit from motion sensors.

3. Introduction to Motion Sensors

Motion Sensors are a class of device which reacts to or senses physical motion parameters, including acceleration, rate, or distance. Inertial sensors are a special class of device which reacts to motion of the sensor itself. Also, motion sensors may indirectly measure motion by magnetic field changes, or changes in pressure. A typical high end cell phone includes Gyroscopes, Accelerometers, magnetic sensors, and pressure sensors to sense device motion [1].

Gyroscope Sensor: This sensor measures angular rate, usually expressed in degrees per second. Integrating angular rate with respect to time results in a measured angle of travel, which can be used to track changes in orientation. Gyroscope sensors are available from a variety of suppliers in single, double or triple axes which correspond to simultaneous measurement of pitch, or roll, or yaw angles. Gyroscopes track relative movement independently from gravity, so errors in bias estimation or integration result in an inherent error, or "drift" [2].

Accelerometer Sensor: This sensor measures acceleration, which includes acceleration components caused by device motion and acceleration due to gravity. The acceleration is measured in G (gees) which are multiples of the earth's gravitational force (1G = 9.8 m/s^2). Accelerometers are also available with single, double or triple axes, defined in an X, Y, Z coordinate system. The accelerometer measures static device orientation by computing the measured angle of the device, compared to gravitational force. Periods of complex motion of the device can cause calculation of orientation to be difficult, particularly during rapid, complex motion, where the signal includes the summation of linear acceleration, centripetal acceleration, and gravity [3].

Magnetic (Mag) Sensor: Magnetic sensors measure magnetic field, typically in units of microTeslas (uT) or Gauss (100 uT = 1 Gauss). The most common version for mobile electronics is a triple axes Hall Effect magnetometer. The magnitude of the Earth's magnetic field varies between 25 and 65 uT, and in angle of inclination depending on geographic location. For the continental United States, the intensity varies between 45 and 55 uT, at an angle between 50 and 80 degrees. By computing the angle of the detected earth's magnetic field, and comparing that measurement angle to gravity as measured by an accelerometer, it is possible to measure a device's heading with respect to North with a high degree of accuracy [4].

Pressure Sensor: Pressure sensors measure differential or absolute pressure and the units are typically hectopascal (hPa) or milliBar (mbar), which are equivalent. Standard atmospheric pressure (at sea level) is defined as 1,013.25 hPa. Changes in altitude (h_{alt}) result in a change in detected ambient air pressure (p_{sta}) according to the equation: $h_{alt} = \left(1 - \left(\frac{p_{sta}}{1013.2}\right)^{0.190284}\right) \times 145366.45$, and can be used to track vertical motion [5].

4. Introduction to Sensor Fusion

Sensor fusion describes the method to derive a single, high accuracy estimate of device orientation or position, combining the output of various sensors. The mathematical algorithms behind sensor fusion are complex, and not covered in great detail. This document focuses on the output of typical sensor fusion algorithms, which are of interest to a typical software developer. For example, Android OS contains several sensor output and motion outputs which are linked here at the end of the paragraph [6].

Gravity: Gravity refers to the earth gravity excluding the acceleration caused by the user, three dimensional vector, with units in m/s^2, consisting of a relative angle between the device body-frame and gravity vector. An accelerometer can measure Gravity when the device is stationary, or in combination with a Gyroscope during periods of motion. There are many applications which utilize Gravity, including detecting orientation of the device with respect to earth, or games which utilize tilt angle to control an airplane's lift or the steering wheel of a car. Phones with a gyroscope sensor will have a much more accurate Gravity measurement when the device is in use, and a faster reaction time to quick changes in orientation.

Linear Acceleration: This is equivalent of acceleration of the device as measured from the accelerometer, with the gravity vector subtracted from this value. The result is the linear acceleration of the device, which can be utilized to measure the device movement in three dimensional space. The accuracy of this value is dependent on the tracking accuracy of the Gravity vector, and if used for motion integration is typically only useful for short durations. The units are in m/s^2 for android OS.

Orientation (attitude): Orientation is similar to the set of Euler angles of yaw (Azimuth), pitch, and roll, with units in degrees. The Android OS Orientation value uses the following definition repeated here: Azimuth (yaw) is defined as the angle between the magnetic north direction and the y-axis, around the z-axis (0 to 359). 0=North, 90=East, 180=South, 270=West. Pitch is defined as the rotation around x-axis (-180 to 180), with positive values when the z-axis moves toward the y-axis. Roll is defined as the rotation around y-axis (-90 to 90), with positive values when the x-axis moves toward the z-axis. The Orientation angles follow the device motion, and can be used for augmented reality, pointing, or aiming applications [6].

Rotation Vector: A Rotation Vector is the result of a sensor fusion algorithm and is derived from a combination of sensor data from accelerometer, gyroscope and magnetometer. The rotation vector represents a rotation angle around a specified axis and corresponds to the vector components of a unit quaternion. In Android OS, if the Rotation Vector value.length = 4, the Rotation Vector is equivalent to a unit quaternion. Android OS includes a helper function, getQuaternionFromVector(), which does this conversion as needed. This function can be used to directly follow device motion. The device Rotation Vector and Quaternion orientation are synonymous [7].

For graphic applications, multiply/divide operations on the device Quaternion are the simplest implementation of motion. A quaternion can be directly manipulated to reflect yaw, pitch and roll movements, and, if needed, reset to the identity quaternion to transfer body-frame orientation to world-frame orientation or other reference orientation. The derivative of a quaternion is also related to angular rate, and can be utilized for various applications [8].

Sensor Fusion Basics: While there are many techniques to perform sensor fusion, this section will describe the basic steps required for a simple form of sensor fusion. The goal is to calculate a device Quaternion from where mathematically the orientation, gravity, rotation vector, rotation matrix, and Euler angle can be derived.

Step 1: Convert Gyroscope angular rate to a quaternion representation, where $\omega(t)$is the angular rate and $q(t)$is the normalized quaternion.

$$\frac{dq(t)}{dt} = \frac{1}{2}\omega(t) * q(t)$$

Step 2: Convert Accelerometer data to world coordinates. This means using the Quaternion above to get the appropriate coordinate system for world-frame motion. Here $A_b(t)$ is the body coordinates of the device, while $A_W(t)$ is in world-frame.

$$A_W(t) = q(t) * A_b(t) * q(t)'$$

Step 3: Create acceleration measurement feedback quaternion as below.

$$qf(t) = \begin{bmatrix} 0 & A_{wy}(t) - A_{wx}(t) & 0 \end{bmatrix} * gain$$

Step 4: Once converted to world coordinates, accel feedback and *gain* is used to generate a feedback quaternion which is then added to previous quat along with gyro generated quaternion. The result is a Quaternion that will track the gyroscope measured data, but will drift towards the accelerometer measurement, according to the value chosen for *gain*.

Similarly, magnetometer data can be added to the yaw component of the quaternion.

Calibration: Sensors exhibit changes in their measurement output over time and require initial (factory) calibration and periodic calibration to maintain performance. Bias and gain, which define the sensor's linear transformation between measurement and resulting value, are subject to drift and instability according to sensor behavior, misalignment, or environmental factors, such as temperature. In Android OS, calibration is performed by the background sensor framework, and the application can simply utilize the pre-calibrated values.

5. Motion Interface Applications

Here is a partial listing of subcategories and descriptions of the types of Motion Interface Applications[9]:

Viewing Applications

a. Panorama Viewer: An application which allows the user to view a panoramic image, giving a 360 degree or wide angle view of an image.

b. Gaming: Immersive gaming where the user can move the device to match the character / object movement, similar to a “First person shooter.”

c. Control: An application where the user controls a toy or device with motion rather than traditional remote joystick, touch or buttons.

d. Virtual Reality: Virtual reality application where a device can be used as a manipulator inside a virtual room or space.

e. LBS: Applications which use motion to detect heading direction and interface with , often in combination with other sensors like GPS and WIFI, to track user movement to enable Location Based Services.

f. Gallery: Interface with an image gallery or menu in 3D space.

Image Capture Applications

a. Panorama Capture: Applications which use motion tracking to assist in taking panoramic or 360 degree images with a device camera.

b. Interface with camera to mapping like regular maps, indoor maps, mapping a room

c. 3D: Applications which use motion tracking to assist in taking a 3D view of an object, through a sequence of still photos or video frames.

d. Image stabilization: Using device motion to assist, track, correct, or inform a user in taking higher quality images.

e. Video Stabilization: Implementations of video stabilization using motion data to eliminate rolling shutter effects or other artifacts during video capture.

User interface

a. Gestures: Gesture assisted user interface where normal gesture like shake, swipe, flick can be included as a user interface feature instead of a button or touch

b. Smart TV: Motion based interface to use it as pointing device for TV’s etc.

Activity and sports

a. Activity Monitor: Use to detect human activity and link it with health and wellness programs, including calorie counting.

b. Pedometer: Applications which track steps.

c. Self-Improvement: Applications which utilize the motion sensor of the device to track sleep habits, posture, walking gait, etc., with the intent on providing information for healthy living.

6. Reference

[1] Sensors link http://developer.android.com/reference/android/hardware/SensorEvent.html
[2] Wiki Gyroscope related link http://en.wikipedia.org/wiki/Gyroscope
[3] Wiki Accelerometer related link http://en.wikipedia.org/wiki/Accelerometer
[4] Wiki Magnetometer related link http://en.wikipedia.org/wiki/Magnetometer
[5] Wiki pressure sensor related link http://en.wikipedia.org/wiki/Pressure_sensor
[6] Android sensor outputs http://developer.android.com/reference/android/hardware/SensorEvent.html
[7] Quaternion fundamentals http://en.wikipedia.org/wiki/Quaternion
[8] Sensor Fusion gyro integration http://www.euclideanspace.com/physics/kinematics/angularvelocity/
[9] Applications http://www.invensense.com/mems/applications.html
[10] J. Diebel, *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors.* http://www.swarthmore.edu/NatSci/mzucker1/e27/diebel2006attitude.pdf.

Appendix: Different Representations of Device Orientation

The device orientation, or attitude, can be represented in many different mathematic forms, such as Euler angles, quaternion, rotation matrix, etc. In this paper we will give an overview to these different forms of representation and also provide a simple explanation to show how to derive the representation from one form to another.

Device Orientation: World Coordinates and Device Coordinates

Figure 1 shows the common definition of the coordinate systems used in handheld devices. The device coordinate system (the left image) is defined relative to the screen of the phone in its default orientation. The X axis is horizontal and points to the right of the device, the Y axis is vertical and points up, and the Z axis points towards the outside of the front face of the screen. In this system, coordinates behind the screen have negative Z values. In the world coordinate system (the right image), the Z axis is perpendicular to the ground, and thus the negative Z points to the direction of the earth gravity. The X axis roughly points to East and the Y axis points to magnetic North.

To create a motion-aware phone applications (e.g. panorama viewer), we would like to keep track of the orientation of the device toward the world coordinate system at all times.

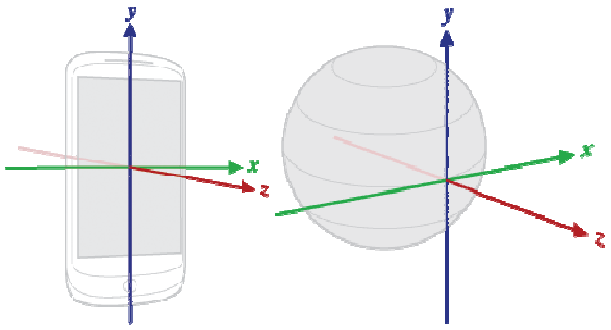

Figure 1. Device coordinate system (left) and world coordinate system (right).

Euler Angles

The **Euler angles** are three angles introduced by Leonhard Euler (1707-1783) to describe the orientation of a rigid body. Euler angles are commonly used in navigation and robotics to describe aircraft attitude or robotic arm movement. Euler's rotation theorem tells us that any orientation can be described as three consecutive rotations. The following illustration shows a sequence of the three consecutive rotations that changes the orientation of a device from the coordinate system **xyz** to **x'y'z'**:

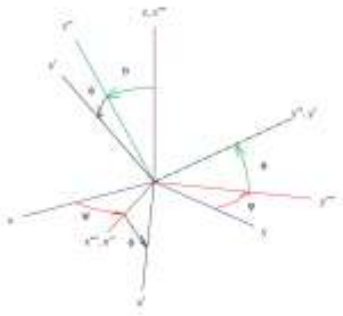

Figure 2. Z-X'-Z' Euler angle definition.

1. **xyz** to **x'''y'''z'''** (**yaw**): rotate by an angle ψ along the **z**-axis,
2. **x'''y'''z'''** to **x''y''z''** (**pitch**): rotate by an angle Θ along the former x-axis (now **x'''**),
3. **x''y''z''** to **x'y'z'** (**roll**): rotate by an angle φ along the former y-axis (now **y''**).

The Euler angles (Θ, φ, ψ) are also called "**pitch**", "**roll**", and "**yaw**" angles. We can use the phone in Figure 1 to explain the rotation sequence. Suppose we want to program a robotic arm to move the phone to a certain orientation from the phone's initial position. The robotic arm can complete the task by running the following three steps of action:

1. rotate the phone by an angle ψ along the z axis while keep the phone on the flat surface (i.e. yaw movement),
2. raise the upper half of the phone by an angle Θ (i.e. pitch movement),
3. rotate the phone along its long axis by an angle φ (i.e. roll movement).

Note that the rotation sequence is non-commutative. Shuffling the order may result in different orientation.

Also note that the Euler angle definition is not "unique". There are many different conventions for Euler angles, depending on the axes along which the rotations are carried out. The above rotation sequence (Z, X, and then Y) is called the *(2,1,3) sequence*. For simplicity sake, in the rest of this paper we will use only the (2,1,3) sequence to discuss the relations between different rotation representations. The other Euler conventions can be derived in a similar matter. For a complete explanation of all the Euler conventions please refer to [10].

Rotation Matrix

The rotation matrix is another way to describe orientation. Let's use a simple 2D case to explain the concept. Suppose we have a vector $\mathbf{v}_0 = [1,0]^T$ and its orientation changes by an angle Θ:

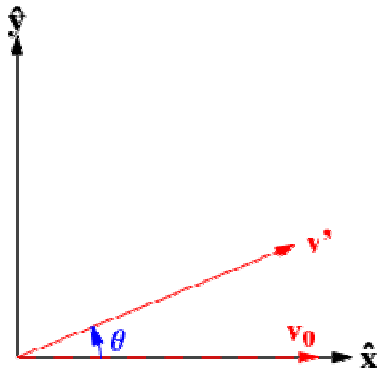

Figure 3. 2D rotation example.

The new orientation **v'** can be obtained by multiplying the vector $\mathbf{v}_0$ with a 2x2 rotation matrix $\mathbf{R}_\theta$:

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{v}' = \mathbf{R}_\theta \mathbf{v}_0 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

To generalize the concept to 3D, we can think of the rotation Θ as a rotation along the z-axis and its 3x3 rotation matrix is

$$\mathbf{R}_{\boldsymbol{\theta}} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The vector $\mathbf{v}_0 = [1,0,0]^T$ after rotation becomes $\mathbf{v'} = [\cos\Theta, \sin\Theta, 0]^T$.

Properties of Rotation Matrix

Rotation matrices are always square, with real entries. Algebraically, a rotation matrix in *n*-dimensions is an $n \times n$ special orthogonal matrix, i.e. an orthogonal matrix whose determinant is 1, and whose inverse matrix is its own transpose.

Multiplying the inverse of a rotation matrix will reverse the rotation. In the example of Figure 3,

$$\mathbf{R}_{\boldsymbol{\theta}}^{-1} = \mathbf{R}_{\boldsymbol{\theta}}^{T} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v_0} = \mathbf{R}_{\boldsymbol{\theta}}^{-1}\mathbf{v}' = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Euler Angles <=> Rotation Matrix

Let's go back to the example in Figure 2 to explain the relation between rotation matrix and Euler angles. In Figure 2, each rotation (yaw, pitch, and then roll) can be written in rotation matrix form:

$$\mathbf{R}_{\boldsymbol{\psi}} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ (yaw, rotate along z-axis)

$$\mathbf{R}_{\boldsymbol{\theta}} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$ (pitch, rotate along y'''-axis)

$$\mathbf{R}_{\boldsymbol{\phi}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$ (roll, rotate along x''-axis)

The combined rotation matrix A is written as

$$\mathbf{A} = \mathbf{R}_{\boldsymbol{\phi}}\mathbf{R}_{\boldsymbol{\theta}}\mathbf{R}_{\boldsymbol{\psi}}$$

In three-dimensional space the rotation matrix is in a 3x3 form:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{23} & a_{33} \end{bmatrix}$$

Below are the elements in the **A** matrix, obtained by multiplying $\mathbf{R}_{\boldsymbol{\phi}}$, $\mathbf{R}_{\boldsymbol{\Theta}}$, and $\mathbf{R}_{\boldsymbol{\psi}}$:

$$a_{11} = \cos\phi\cos\psi - \sin\phi\sin\theta\sin\psi$$
$$a_{12} = \cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi$$
$$a_{13} = -\cos\theta\sin\phi$$
$$a_{21} = -\cos\theta\sin\psi$$
$$a_{22} = \cos\theta\cos\psi$$
$$a_{23} = \sin\theta$$
$$a_{31} = \sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi$$
$$a_{32} = \sin\phi\sin\psi - \cos\phi\sin\theta\cos\psi$$
$$a_{33} = \cos\theta\cos\phi$$

Deriving Euler angles from rotation matrix is straightforward:

$$\phi = atan2(-a_{13}, a_{33})$$
$$\theta = asin(a_{23})$$
$$\psi = atan2(-a_{21}, a_{22})$$

Rotation Matrix => Gravity

We can obtain the direction of earth gravity (relative to the device coordinates) from the rotation matrix. Let's revisit the example in Figure 2. Suppose at the beginning the device coordinates are aligned with the world coordinates, that is, the device is placed on a flat surface with the screen facing up and the top pointing north (Figure 1). In this position the device Z axis aligns with the earth gravity. When the device changes its orientation with rotation matrix **A**, the device Z axis moves from **z** to **z'**,

$$\mathbf{z}' = \mathbf{A}\mathbf{z}$$

Now we need to find the gravity direction **z** relative to the device, i.e. in the device coordinate system. We know that in the device coordinate system the **z'** vector is always $[0,0,1]^T$. Therefore the vector **z** can be obtained by applying the inverse of the rotation matrix,

$$\mathbf{z} = \mathbf{A}^{-1}\mathbf{z}' = \mathbf{A}^{\mathrm{T}}\mathbf{z}' = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{23} & a_{33} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{31} \\ a_{32} \\ a_{33} \end{bmatrix}$$

In other words, the earth gravity vector in device coordinate system can be represented by the bottom row of the rotation matrix **A**.

Quaternion

Euler's rotation theorem also tells us that any orientation can be expressed as a single rotation about some axis. The axis is the unit vector (unique except for sign) which remains unchanged by the rotation (called the Euler axis). The magnitude of the angle is unique, with its sign being determined by the sign of the rotation axis. The axis can be represented as a three-dimensional unit vector $\hat{\mathbf{e}} = [e_x \ e_y \ e_z]^T$, and the angle by a scalar **α**.

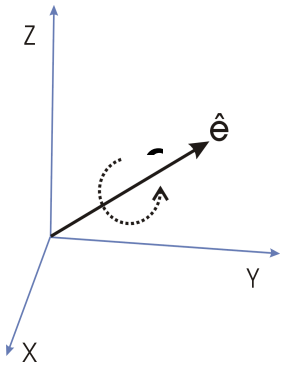

Figure 4. Euler axis and angle.

The Euler axis is well defined with the exception that when **α=0** (if there is no rotation, any direction can be considered as the rotation axis). In 1843, Sir William Hamilton introduced a similar but mathematically superior axial rotation representation, called the ***quaternion***. The quaternion is a 4-element vector $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$, where

$$q_0 = \cos\frac{\boldsymbol{\alpha}}{2}$$
$$q_1 = e_x \sin\frac{\boldsymbol{\alpha}}{2}$$
$$q_2 = e_y \sin\frac{\boldsymbol{\alpha}}{2}$$
$$q_3 = e_z \sin\frac{\boldsymbol{\alpha}}{2}$$
$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

When **α=0**, $\mathbf{q} = [1,0,0,0]^T$.

Quaternion has proven a very useful representation of orientation in many applications such as computer graphics and gaming. To help us understand quaternion's relation with other orientation representations, let us review a few basics of Quaternion Algebra:

The ***conjugate*** of a quaternion **q** is $\mathbf{q}^{-1} = [q_0, -q_1, -q_2, -q_3]^T$, which is the same rotation angle but along the opposite Euler axis. The ***product*** of two quaternions is defined as

$$\mathbf{q} \cdot \mathbf{p} = \begin{bmatrix} q_0 & -\mathbf{q}^T \\ \mathbf{q} & q_0\mathbf{I}_3 + \breve{\mathbf{q}} \end{bmatrix} \begin{bmatrix} p_0 \\ \mathbf{p} \end{bmatrix} = \mathbf{Q} \begin{bmatrix} p_0 \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$\mathbf{p} \cdot \mathbf{q} = \begin{bmatrix} q_0 & -\mathbf{q}^T \\ \mathbf{q} & q_0\mathbf{I}_3 - \breve{\mathbf{q}} \end{bmatrix} \begin{bmatrix} p_0 \\ \mathbf{p} \end{bmatrix} = \bar{\mathbf{Q}} \begin{bmatrix} p_0 \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

where the skew-symmetric matrix $\breve{\mathbf{q}} = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}$ is the matrix-vector notation for the vector cross product. The quaternion matrices $\mathbf{Q}$ and $\mathbf{P}$ commute, i.e. $\mathbf{Q}\bar{\mathbf{P}} = \bar{\mathbf{P}}\mathbf{Q}$. The matrices of the conjugate quaternion $\mathbf{q}^{-1}$ are transpose of , i.e. $\mathbf{Q}^{\mathbf{T}}$ and $\bar{\mathbf{Q}}^{\mathbf{T}}$.

To represent a point $\boldsymbol{z}$ in 3D space by the quaternion, we form a 4-element vector $\boldsymbol{Z}$ = [0, $\boldsymbol{z}^{T}$]^{T}. To apply rotation on this vector, the quaternion rotation uses the following formula:

$$\boldsymbol{Z}' = \mathbf{q} \cdot \boldsymbol{Z} \cdot \mathbf{q}^{-1}$$

To apply a sequence of rotations, we can concatenate the quaternions:

$$\boldsymbol{Z}' = \mathbf{q_2} \cdot (\mathbf{q_1} \cdot \boldsymbol{Z} \cdot \mathbf{q_1^{-1}}) \cdot \mathbf{q_2^{-1}} = (\mathbf{q_2} \cdot \mathbf{q_1}) \cdot \boldsymbol{Z} \cdot (\mathbf{q_2} \cdot \mathbf{q_1})^{-1}$$

Quaternion <=> Euler Angles

The quaternion from a particular Euler sequence can be written as the product of three quaternions:

$\mathbf{q_A} = \mathbf{q_\phi} \cdot \mathbf{q_\theta} \cdot \mathbf{q_\psi}$

where $\mathbf{q_\psi} = \begin{bmatrix} \cos\frac{\psi}{2} \\ 0 \\ 0 \\ \sin\frac{\psi}{2} \end{bmatrix}, \mathbf{q_\theta} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} \\ 0 \\ 0 \end{bmatrix}, \mathbf{q_\phi} = \begin{bmatrix} \cos\frac{\phi}{2} \\ 0 \\ \sin\frac{\phi}{2} \\ 0 \end{bmatrix}$.

After quaternion multiplications (details skipped here), the final quaternion is

$$\mathbf{q_A} = \begin{bmatrix} \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} - \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \end{bmatrix}$$

The inverse mapping (from quaternion to Euler angles) can be obtained by rearranging the equation above:

$$\boldsymbol{\phi} = \boldsymbol{atan2}(-2q_1q_3 + 2q_0q_2, q_3^2 - q_2^2 - q_1^2 + q_0^2)$$
$$\boldsymbol{\theta} = \boldsymbol{asin}(2q_2q_3 + 2q_0q_1)$$

$$\psi = \boldsymbol{atan2}(-2q_1q_2 + 2q_0q_3, q_2^2 - q_3^2 - q_1^2 + q_0^2)$$

Quaternion <=> Rotation Matrix

Rotation matrix can be derived from quaternion with the following. To rotate a vector ***z*** (whose quaternion representation is $\boldsymbol{Z} = [0, \boldsymbol{z}^T]^T$), the quaternion rotation formula is this:

$$\boldsymbol{Z}' = \mathbf{q} \cdot \boldsymbol{Z} \cdot \mathbf{q}^{-1}$$

$$\begin{bmatrix} 0 \\ \boldsymbol{z}' \end{bmatrix} = (\mathbf{Q}\begin{bmatrix} 0 \\ \boldsymbol{z} \end{bmatrix}) \cdot \mathbf{q}^{-1}$$

$$= \bar{\mathbf{Q}}^{\mathbf{T}}\mathbf{Q}\begin{bmatrix} 0 \\ \boldsymbol{z} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{1} & \mathbf{0}^{\mathbf{T}} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}\begin{bmatrix} 0 \\ \boldsymbol{z} \end{bmatrix}$$

where **R** is the rotation matrix with

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_0q_3 + q_1q_2) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Rearrange the above equation and we can also compute quaternion from rotation matrix:

$$q_0 = \pm\frac{1}{2}\sqrt{1 + a_{11} + a_{22} + a_{33}}$$

$$q_1 = \frac{1}{4q_0}(a_{23} - a_{32})$$

$$q_2 = \frac{1}{4q_0}(a_{31} - a_{13})$$

$$q_3 = \frac{1}{4q_0}(a_{12} - a_{21})$$

Note that the result shows two possible solutions, with one being the negative of the other. Both **q** and **-q** ($= [-q_0, -q_1, -q_2, -q_3]^T$) are valid quaternions since they represent the same orientation (**-q** is Euler axis pointing to opposite direction with (180- **α**) degrees of rotation).